Programming Project 3

STL Stack to convert infix expression to post fix and evaluate

- **Due** Oct 8 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** cpp
- **Available** until Oct 8 at 11:59pm

For this project you will use the STL stack provided with C++ to evaluate convert an infix expression to postfix and evaluate it. You can find the general C++ reference for the STL stack here (Links to an external site.)Links to an external site.. There is some brief sample code here (Links to an external site.)Links to an external site.. Note that where the book uses the peek() method the STL stack has the top() method--they are functionally the same.

Requirements

- Write a program in the file Project3.cpp which prompts the user for an infix expression and uses STL stack objects to convert it to postfix then evaluate it. The program should:

  1. Use a function called validInfix() which returns a Boolean value to determine if the infix expression entered is valid. If the user enters an invalid infix expression, re-prompt them until they enter a valid expression. An infix expression is valid if:
     - There are no characters other than numbers, operators, and parenthesis. So 1 + (3 * 4) is valid, x + (3 * y) is not.
     - Each operator is between 2 operands. 3 * 4 is valid, 3 4 * is not. Note that parenthesis contain operands, so the * in 2 * (1 + 3) is valid because it is between 2 and (.
     - There are the same number of opening and closing parenthesis in the correct order.
     - Note for this project we will assume parenthesis are in the correct location(s) as long as they are matching, though this isn't necessarily true.
  2. Follow the pseudocode algorithm on page 207 to convert the infix expression to postfix using the STL stack. Once done, display the postfix expression with spaces between all operators and operands (see sample output below).
  3. After converting to postifx, use an STL stack to evaluate the postfix expression then display the answer. Follow the pseudocode algorithm on page 204.
  4. Prompt the user to see if they'd like to evaluate another expression.

- You may assume:
  - The infix expression will contain no spaces.
  - The infix expression will use only these operators: +, -, *, and /.
  - The infix expression will contain only integers, operators, and parenthesis (no decimal points). Note that it may contain integers greater than 9 (i.e. 10, 99, 250, etc.) so handle those accordingly.
- Break the program into logical functions, **do not** place all the code in main().

Sample Output

A run of your program should look something like this:

```
Enter an infix expression with no spaces: 2+(3*5)
In postfix this is: 2 3 5 * +
Answer: 17
Would you like to do it again? (Yes/No) No
```

Other requirements

- Follow the course code style guide and good programming practices.
- Make sure your code handles incorrect/unusual data properly.
- If you wrote the program in an environment other than Visual Studio 2017 it is your responsibility to ensure that it works in VS 2017 (which is what I'll be grading in).
- Put a comment at the top of all files with your full name, class number, and assignment name like so:

```
// Joe Smith
// CS250 Programming Project 1
```

Due Date

See below.

Turn in

Upload Project3.cpp. *Do not* upload the entire Visual Studio project.